CLAIMS

What is claimed is:

1. A method comprising:

receiving a first program unit in a parallel computing environment, the first program unit including a memory update operation to be performed atomically, the memory update operation having an operand, the operand being of a data-type and of a data size; and

translating the first program unit into a second program unit, the second program unit to associate the memory update operation with a set of one or more low-level instructions upon determining that the data size of the operand is supported by the set of low-level instructions, the set of low-level instructions to ensure atomicity of the memory update operation.

2. The method of claim 1 wherein to associate the memory update operation with the set of low-level instructions comprises:

enclosing the memory update operation in a callback routine; and

referencing the callback routine from a routine that references the set of low-level instructions.

3. The method of claim 1 wherein the set of low-level instructions encapsulate the memory update operation.

4. The method of claim 1 further comprising translating the first program unit into a third program unit upon determining that a second set of one or more low-level

instructions support the memory update operation for the data-type and the data size of the operand, the second set of low-level instructions for performing the memory update operation atomically.

5. The method of claim 1 further comprising translating the first program unit into a third program unit upon determining that a second set of low-level instructions does not support the memory update operation for the data-type and the data size of the operand and that the set of low-level instructions does not support the data size of the operand, the third program unit to associate the memory update operation with a set of locking instructions.

6. The method of claim 1 wherein the set of low-level instructions for ensuring atomicity is a compare-and-swap instruction.

7. A method comprising:

receiving a first program unit, the first program unit including a memory update operation to be performed atomically, the memory update operation indicating an operand and an operator, the operand being of a data-type and a data size;

translating the first program unit into a second program unit upon determining that a first set of one or more low-level instructions support the memory update operation for the data-type and the data size of the operand, the first set of low-level instructions for performing the memory update operation atomically;

translating the first program unit into a third program unit, the third program unit to associate the memory update operation with a second set of one or more low-level instructions upon determining that the data size of the operand is supported by the second set of low-level instructions, the second set of low-level instructions to ensure atomicity of the memory update operation; and

translating the first program unit into a fourth program unit upon determining that the first set of low-level instructions does not support the memory update operation for the data-type and the data size of the operand and that the second set of low-level instructions does not support the data size of the operand, the fourth program unit to associate the memory update operation with a set of locking instructions.

8. The method of claim 7 wherein the second set of low-level instructions encapsulate the memory update operation.

9. The method of claim 7 wherein associating the second set of instructions to the memory update operation comprises:

enclosing the memory update operation in a callback routine; and

referencing the callback routine from a routine that references the second set of low-level instructions.

10. The method of claim 7 wherein the second set of low-level instructions is a compare-and-swap instruction.